STATUS OF THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Previously Presented) A method of compiling a program comprising:

identifying a set of speculative parallel thread candidates;

determining a misspeculation cost value for at least one of the speculative parallel thread candidates;

selecting a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value; and

generating program code based on the set of speculative parallel threads.

2. (Original) A method as defined in claim 1 wherein identifying the set of speculative parallel thread candidates comprises identifying program regions.

3. (Original) A method as defined in claim 1 wherein at least one of the speculative parallel thread candidates comprises at least one program region.

4. (Original) A method as defined in claim 1 wherein at least one of the speculative parallel threads comprises at least one program region.

5. (Original) A method as defined in claim 1 wherein identifying the set of speculative parallel thread candidates comprises identifying program loops.

6. (Original) A method as defined in claim 1 wherein at least one of the speculative parallel thread candidates comprises a program loop.

7. (Original) A method as defined in claim 1 wherein at least one of the speculative parallel threads comprises a program loop.

8. (Original) A method as defined in claim 1 wherein identifying the set of speculative parallel thread candidates comprises identifying a main thread.

9. (Original) A method as defined in claim 8 wherein the main thread comprises a current iteration of a program loop, and the speculative parallel thread candidate comprises a next iteration of the same program loop.

10. (Original) A method as defined in claim 8 wherein the main thread comprises a current iteration of a program loop, and the speculative parallel thread comprises a next iteration of the same program loop.

11. (Canceled)

12. (Previously Presented) A method as defined in claim 1 wherein determining the misspeculation cost value comprises:

identifying a data dependency in at least one of the speculative parallel thread candidates;

determining, for the data dependency, a likelihood that a dependency violation will occur; and

determining an amount of computation required to recover from the data dependency violation.

13. (Original) A method as defined in claim 1 further comprising determining at least one of the following for at least one of the speculative parallel thread candidates:

a size of the speculative parallel thread candidate; and

a likelihood representative of the speculative parallel thread candidate.

14. (Previously Presented) A method as defined in claim 1 wherein at least one of the speculative parallel thread candidates is transformed prior to determining the misspeculation cost value for the at least one of the speculative parallel thread candidates.

15. (Original) A method as defined in claim 14 wherein the at least one of the speculative parallel thread candidates is transformed by a code reordering.

16. (Original) A method as defined in claim 14 further comprising determining at least one of the following for at least one of the speculative parallel thread candidates:

a size of the speculative parallel thread candidate;

a likelihood representative of the speculative parallel thread candidate; and

a description of the transformation performed on the speculative parallel thread candidate.

17. (Original) A method as defined in claim 1 wherein at least one of the speculative parallel threads is transformed prior to code generation.

18. (Original) A method as described in claim 17 wherein the at least one of the speculative parallel threads is transformed by code reordering.

19. (Previously Presented) An article of manufacture storing machine readable instructions that, when executed, cause a machine to:

identify a set of speculative parallel thread candidates;

determine a misspeculation cost value for at least one of the speculative parallel thread candidates;

select a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value; and

generate program code based on the set of speculative parallel threads.

20. (Canceled)

21. (Previously Presented) An article of manufacture as defined in claim 19 wherein, to determine the misspeculation cost value, the machine readable instructions cause the machine to:

identify a data dependency in at least one of the speculative parallel thread candidates;

determine, for the data dependency, a likelihood that a dependency violation will occur; and

determine an amount of computation required to recover from the data dependency violation.

22. (Original) An article of manufacture as defined in claim 19 wherein the machine readable instructions cause the machine to determine at least one of the following for at least one of the speculative parallel thread candidates:

a size of the speculative parallel thread candidate; and

a likelihood representative of the speculative parallel thread candidate.

23. (Previously Presented) An article of manufacture as defined in claim 19 wherein the machine readable instructions cause the machine to transform at least one of the speculative parallel thread candidates prior to determining the misspeculation cost value.

24. (Previously Presented) An apparatus to compile a program comprising:

a candidate identifier to identify a set of speculative parallel thread candidates;

a metric estimator to determine a misspeculation cost value for at least one of the speculative parallel thread candidates;

a speculative parallel thread selector to select a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value; and

a code generator to generate program code based on the set of speculative parallel threads.

25. (Original) An apparatus as defined in claim 24 wherein the candidate identifier comprises a region identifier to identify program regions.

26. (Original) An apparatus as defined in claim 24 wherein the candidate identifier comprises a loop identifier to identify program loops.

27. (Original) An apparatus as defined in claim 24 wherein the candidate identifier comprises a candidate selector to select a first one of a program region and a program loop iteration to execute in a main thread, and to select a second one of a program region and a program loop iteration to execute in a speculative parallel thread.

28. (Canceled)

29. (Previously Presented) An apparatus as defined in claim 24 wherein the metric estimator comprises:

a data dependency identifier to identify a data dependency in at least one of the speculative parallel thread candidates;

a likelihood evaluator to determine a likelihood that a dependency violation will occur; and

a recovery size calculator to determine an amount of computation required to recover from the data dependency violation.

30. (Original) An apparatus as defined in claim 24 wherein the candidate identifier determines at least one of the following for at least one of the speculative parallel thread candidates:

a size of the speculative parallel thread candidate; and

a likelihood representative of the speculative parallel thread candidate.

31. (Previously Presented) A system to compile a program comprising:

a candidate identifier to identify a set of speculative parallel thread candidates;

a metric estimator to determine a misspeculation cost value for at least one of the speculative parallel thread candidates;

a speculative parallel thread selector to select a set of speculative parallel threads from the set of speculative parallel thread candidates based on the misspeculation cost value;

a code generator to generate program code based on the set of speculative parallel threads; and

a static random access memory to store the misspeculation cost value.

32. (Previously Presented) A system as defined in claim 31 wherein the metric estimator comprises:

a data dependency identifier to identify a data value dependency in at least one of the speculative parallel thread candidates;

a likelihood evaluator to determine a likelihood that a dependency violation will occur; and

a recovery size calculator to determine a set of recovery computation sizes that represent an amount of computation required to recover from the data dependency violation.

33. (Previously Presented) A method of compiling a program comprising:

identifying a data dependency violation in a set of speculative parallel thread candidates;

determining a likelihood that the data dependency violation will occur;

determining an amount of computation required to recover from the data dependency; and

selecting at least one of the set of speculative parallel thread candidates based on a lowest likelihood of misspeculation.